# Preprocess your Paths – Speeding up Linear Programming-based Optimization for Segment Routing Traffic Engineering

Alexander Brundiers°, Timmy Schüller•, Nils Aschenbruck°

°Osnabrück University, Institute of Computer Science, Germany
Email: {brundiers, aschenbruck}@uos.de

•Deutsche Telekom Technik GmbH, Germany
Email: timmy.schueller@telekom.de

***Abstract*—Many state-of-the-art Segment Routing (SR) Traffic Engineering (TE) algorithms rely on Linear Program (LP)-based optimization. However, the poor scalability of the latter and the resulting high computation times impose severe restrictions on the practical usability of such approaches for many use cases. To tackle this problem, a variety of preprocessing approaches have been proposed that aim to reduce computational complexity by preemptively limiting the number of SR paths to consider during optimization. In this paper, we provide the first extensive literature review of existing preprocessing approaches for SR. We further conduct a large-scale comparative study of these approaches using various real-world topologies, including recent data from a Tier-1 Internet Service Provider (ISP) backbone. Based on the insights obtained from this evaluation, we finally propose a combination of multiple preprocessing approaches and show that applying such a preprocessing prior to optimization can reduce the number of 2SR paths to consider by as much as 97-99%, while still achieving close to optimal solutions. We further demonstrate that this allows to reliably reduce computation times of different LP-based TE algorithms by around a factor of 10 or more, without resulting in a relevant deterioration of the solution quality. This is a major improvement over the current state-of-the-art and facilitates the reliable usability of LP-based optimization for large segment-routed networks.**

***Index Terms*—traffic engineering, segment routing, optimization, performance**

## I. Introduction

Segment Routing (SR) has become a premier choice for Traffic Engineering (TE) purposes in large networks. It offers great traffic steering capabilities while simultaneously offering good scalability. However, in order to use SR to its full potential, optimization algorithms are needed to compute the best possible TE configurations. In many state-of-the-art approaches (e.g., [1], [2], or [20]), this is done using Linear Program (LP)-based optimization because it can provide guaranteed optimal solutions. Its major drawback, however, is its limited scalability and the resulting high computation times for larger networks. Depending on the algorithm and the size of the network, those can reach up to multiple hours or even days. For certain use cases, this is acceptable but, for many scenarios, such high computation times severely limit the practical usability of LP-based SR TE algorithms.

Over the recent years, a variety of preprocessing approaches have been proposed that aim to reduce the problem complexity and, thus, the resulting computation time by preemptively limiting the number of SR paths to consider during optimization. While the individually reported results for those approaches look promising, evaluations are often carried out on a rather limited set of data and varying hardware. This raises questions regarding the generalizability of the results and makes it virtually impossible (i.e. for operators) to compare approaches against each other to select the best fitting one.

To address these problems, we provide an extensive literature review and discussion of existing preprocessing approaches to then carry out a large comparative study regarding their performance. For this, we not only use various publicly available topologies from the Repetita dataset [11] but also recent network data from the backbone of a globally operating Tier-1 Internet Service Provider (ISP). Finally, based on insights gained from this evaluation, we propose a combination of multiple preprocessing approaches and show that this leads to a significant improvement in performance. It allows for a reduction of computation times by a factor of 10 or more without a practically relevant deterioration in solution quality. This is a major improvement over the current state-of-the-art and an important step towards the reliable usability of LP-based SR TE for large networks.

The remainder of this paper is structured as follows. First, a general introduction to the topic of SR is given in Section II, followed by a literature review and on existing SR preprocessing approaches (Section III). After this, Section IV describes our evaluation setup, focusing on the used datasets and considered algorithms. The results of the respective evaluation are presented and discussed in Section V. Our approach for a combined preprocessing in proposed and evaluated in Section VI, followed by a discussion of possible limitations of our study (Section VII). Finally, the paper is concluded in Section VIII with a recapitulation on its most important contributions and findings and an outlook on possible future research directions.

## II. An Introduction to Segment Routing

SR [9] is a network tunneling technique that implements the source routing paradigm. Its key feature is the possibility to add specific labels (also called *segments*) to a packet, which function as waypoints that the packet has to visit in a given


DOI: https://doi.org/10.23919/IFIPNetworking62109.2024.10619747

$$\min\ \theta \tag{1}$$

$$\text{s.t.} \quad \sum_k x_{ij}^k = 1 \qquad \forall ij \tag{2}$$

$$\sum_{ij} t_{ij} \sum_k g_{ij}^k(e) x_{ij}^k \le \theta\, c(e) \qquad \forall e \tag{3}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall ijk \tag{4}$$

Problem 1: 2SR formulation (inspired by [1]).

order before heading to its original destination. Depending on the nature of the related waypoint, different segment-types are used. For example, *node segments* refer to routers, while *adjacency segments* identify individual links. The forwarding paths between the waypoints are determined by the Interior Gateway Protocol (IGP) of the respective network. Overall, SR enables the definition of virtually arbitrary forwarding paths and allows for a precise, per-flow traffic control. For this reason, SR has become one of the premier choices for TE and there is a large body of work regarding SR in general and its applications for TE in particular (cf. e.g., [27]).

One of the fundamental works in the SR TE landscape is [1]. Here, the authors propose an LP-based optimization model that builds the foundation for many subsequent works. With it, they show that even SR with just two node segments (2SR) already enables virtually optimal TE solutions in many scenarios. A slightly adapted version of the respective 2SR LP formulation is shown in Problem 1. The objective is to minimize the Maximum Link Utilization (MLU) denoted by $\theta$. The variables $x_{ij}^k$ indicate the percentage share of the demand $t_{ij}$ between nodes $i$ and $j$, that is routed over the intermediate segment $k$. Equation (2) ensures that each demand is satisfied. Equation (3), is the so called *capacity constraint*. For every edge $e$, $g_{ij}^k(e)$ indicates the load that is put on $e$ if a uniform demand is routed from $i$ to $j$ over the intermediate segment $k$. These values are constants and can be efficiently precomputed. All in all, the left side of the constraint denotes the traffic that is put on $e$ by the SR configuration represented by the $x_{ij}^k$. This is then limited to the edges capacity $c(e)$ scaled by $\theta$. By minimizing this scaling factor, a SR configuration with minimal MLU is computed. The only difference to the original LP of [1] is that there the $x_{ij}^k$ variables were continuous, allowing for demands to be split arbitrarily across various SR paths. However, such an arbitrary splitting is not feasible in practice [20]. Therefore, newer variations of the 2SR LP generally prohibit splitting demands over multiple SR paths by making the $x_{ij}^k$ binary variables (cf. e.g., [7] or [20]).

A recent innovation in the SR landscape is the *Midpoint Optimization (MO)* concept [2], [6]. Demands no longer have to be optimized individually by deploying dedicated end-to-end SR tunnels. Instead, a single SR tunnel can be used to detour a whole set of demands. MO allows for a substantial reduction of the number of SR tunnels that need to be deployed to implement TE solutions, lowering the configuration effort and overhead in the network. However, the underlying optimization problem becomes inherently more complex, resulting in substantially higher computation times.

## III. Preprocessing Approaches for LP-based Segment Routing Optimization

A major challenge in the context of SR TE is the scalability of the used optimization algorithms. While LP-based approaches offer the major advantage of providing provable optimal solutions, they scale rather poorly with network size. For small to medium sized networks, this is no issue since solutions can still be computed within seconds or at most minutes. However, for large networks (e.g., WANs or ISP backbones), computing TE solutions with LPs can take multiple hours or more (cf. [20]), while, in practice, solutions might be needed on a timescale of just a few minutes (cf. [12], [13]). There are different ways to approach these issues. Some focus on the use of advanced mathematical concepts like *column generation* [15] or *constraint programming* [12], while others try to deploy meta-heuristics to compute reasonable good solutions within really short timespans (e.g., [3] or [10]).

A completely different approach to bring down the complexity and, hence, computation time of SR LPs focuses on *preprocessing* the set of SR paths to consider during optimization. Each SR path basically consists of the source and destination node of the packet as well as a set of *middlepoints*[1] (the node segments) that it has to visit (cf. Section II). In basically all SR LP formulations (e.g., in Problem 1), the model allows for every node segment to be used as a middlepoint for each traffic demand (aka. source-destination pair). While this guarantees optimality, it also is responsible for a large portion of the overall problem complexity. For every demand the optimization has $|V|^{k-1}$ paths to choose from, resulting in a total number of $\mathcal{O}(|V|^{k+1})$ possible SR paths to evaluate, with $|V|$ being the number of nodes in the network and $k$ the maximum number of segments per path. Here, the preprocessing (or also called *middlepoint selection* [26]) approaches come into play and try to reduce this complexity by limiting the set of available middlepoints per demand and, thus, the set of SR paths to consider prior to optimization. This results in smaller and generally faster to solve LPs.

In the following, we provide a detailed overview on (to the best of our knowledge) all existing preprocessing approaches in the SR TE literature.

### A. Centrality-based Approaches

One of the first works that came up with the idea of preemptively limiting the number SR paths that are considered for optimization are [25], [26]. They only allow a certain subset of nodes as *middlepoints* (aka. intermediate segments) for SR paths and propose to use *graph centrality* metrics to select "important" or "central" nodes into this subset. This idea is evaluated in the context of datacenter networks and ISP backbones for various subset sizes and with different centrality measures. It is observed that, out of all considered

[1]The term "*middlepoint*" used in the remainder of this paper is **not** to be confused with the term "*midpoint*" from the MO concept (Section II).

centrality metrics, selecting the allowed middlepoints based on their *Group Shortest-Path (GSP) centrality* [8] performs best. For a group of nodes $\mathcal{G}$ the GSP centrality is defined as:

$$C_{gsp}(\mathcal{G}) = \sum_{s,t \in V | s,t \notin \mathcal{G}} \frac{\theta_{st}(\mathcal{G})}{\theta_{st}} \tag{5}$$

with $\theta_{st}$ denoting the total number of shortest paths from $s$ to $t$ and $\theta_{st}(\mathcal{G})$ being the number of shortest paths from $s$ to $t$ that include any node in $\mathcal{G}$ [26]. In other words, it characterizes how "central" a group of nodes is based on the number of shortest paths that run through this group.

Overall, it is shown that, when focusing on only a small number of nodes as available middlepoints, computation times can be substantially reduced. However, this comes at the prices of a considerable deterioration in solution quality. While the authors argue that this can be a sensible trade-off to make, in practice, a deterioration of solution quality is only acceptable up to a certain point. Furthermore, limiting the available middlepoints to the same small set of nodes for all demands can result in severe violations of certain operational latency constraints (i.e. from service level agreements). For example, if, for a globe-spanning network, the most "central" nodes are all located in Europe, intra-US traffic either needs to always follow its shortest path or be detoured all the way over a node in Europe, most likely exceeding latency bounds. In addition to that, if the number of SR paths grows larger and they are all forced over the same handful of middlepoints, this can put additional burden on the routing hardware of these nodes.

### B. Stretch-Bounding

Another early middlepoint selection approach is the *Stretch-Bounding (SB)* concept proposed in [23], [24]. Its key idea is to rule out all nodes from being a potential middlepoint for an SR path if they are "too far away" from its source or destination regarding a given metric. This can be formalized as only considering middlepoint $m$ for an SR path between $src$ and $dst$ if the following equation is satisfied:

$$\frac{DIST(src \rightarrow m) + DIST(m \rightarrow dst)}{DIST(src \rightarrow dst)} \leq \alpha_{SB} \tag{6}$$

with the $DIST()$ function denoting the shortest path distance between the respective two nodes and $\alpha_{SB} \in [1, \infty]$ being the so called *SB factor*. This approach rules out all those SR paths that are more then $\alpha$-times longer than the shortest path between the respective source and destination. It is shown in [23] that there is a trade-off between speedup and deterioration of solution quality depending on the chosen $\alpha$-value. The authors define a factor of around $\alpha_{SB} = 1.4$ as the sweetspot of achieving close to optimal results while still considerably speeding up computations by a factor of 3 to 4. However, the SB implementation as described in Equation 6 inherits an issue that can negatively impact performance in certain scenarios (first pointed out in [7]). If the initial shortest path length is small (e.g., for paths with just one or two hops), small $\alpha$-values can completely prohibit any kind of detour for the respective demand. The best example for this is a simple hop-count metric. If the shortest path of a demand has length 1 (one hop), this means that for all $\alpha < 2$, there are no detours available for this demand as every detour would have at least length two. This can negatively impact the achievable MLU.

A rather similar concept to the SB approach of [24] was also proposed in [21], where nodes are assigned geographical tags on three different levels of granularity (`site`, `country`, and `continent`). SR paths between nodes that share a common tag value (e.g., *US* for the `country`-tag) are restricted to only use middlepoints with the same tag value (e.g., only nodes also located in the US). This also implements the idea of limiting the length of SR detours to a sensible maximum (e.g., by not routing traffic between Boston and New York over Europe).

### C. Demand Pinning

Another preprocessing technique briefly described by in [17] and [18] is *Demand Pinning (DP)*. It is based on the observation that in many networks (i.e. WANs and ISP backbones) traffic flows are not uniformly distributed in size. Instead, traffic consist of a few very large demands that make up a considerable amount of the total traffic volume and a rather large number of very small demands. DP fixes the forwarding paths of all these small demands to standard Shortest Path Routing (SPR) and only runs a TE optimization for the larger ones. The idea is that the impact of the small demands on the overall solution quality is negligible compared to the larger traffic flows. Not optimizing them will have virtually no impact on the overall solution quality. To the best of our knowledge, there are no studies on the performance of the DP approach regarding its impact on solution quality and computation time. However, in [17], it is mentioned that around 68% of demands in the *Microsoft* WAN are of small size, making up a combined total of only 1.3% of the total traffic volume. If these results transfer to other networks, as well, DP seems like a promising candidate for an SR preprocessing since fixing the path of 68% of demands would translate to an equal reduction in the number of SR paths to consider during optimization.

### D. SR Path Domination

All of the previous approaches carry the risk of excluding SR paths that are needed for an optimal solution. As a result, the solution quality can become arbitrarily worse when deploying these methods. To prevent this, one has to ensure to only exclude SR paths for which it can be proven that they are not needed for an optimal solution. A first step towards such an approach was presented in [4]. There, it is shown that a large portion of configurable SR paths actually contain loop-like structures and the authors suspect that many of these paths are not required to obtain optimal solutions. This assumption is further investigated and confirmed by Callebaut et al. [7]. They propose the concept of *dominated* and *equivalent* SR paths. An SR path $p_1$ is *dominated* by another path $p_2$ if three conditions are satisfied. First, both paths must have the same start- and endpoint. Second, assuming a uniform traffic flow is routed over each path, for each link $l$ in the set $\mathcal{L}(p_2)$ of links used by $p_2$ the load put on $l$ by $p_2$ must be lower or equal to the load put on $l$ by $p_1$:

$$load(l, p_2) \leqslant load(l, p_1) \qquad \forall\, l \in \mathcal{L}(p_2) \tag{7}$$

Lastly, for at least one link in Equation 7 the strict inequality must hold. Analogously, two SR paths are *equivalent*, if their set of used links and the resulting link-loads are exactly equal. This is the case if the first two conditions for SR path domination hold but with exact equality for Equation 7.

Dominated SR paths are never needed for an optimal solution and for a set of equivalent paths, it is sufficient to consider just one of them, allowing to exclude all others. It is shown in [7] that, based on these two observations, a substantial number of SR paths can be ruled out prior to optimization, resulting in a significant reduction in computation time. Furthermore, just like SB and centrality-based preprocessing approaches, this just requires information on the network topology but not on traffic. Hence, it can be precomputed in advance which is quite useful since in [7], computation times of up to 30min or more are reported for just the preprocessing of larger topologies.

### E. Discussion

As described in the previous sections, there is a wide variety of possible preprocessing approaches for SR. However, judging and comparing the quality and usefulness of these different approaches proves to be difficult. The reasons for this are manifold: Meaningful cross comparisons between the publications are virtually impossible as they all use (i) different hardware as well as (ii) varying datasets. Furthermore, (iii) basically all evaluations are carried out on (semi-)artificial data, like the Repetita dataset which features topologies based on real-world networks but related traffic matrices are fully artificial (cf. Section IV-A). Even if full-on real-world data[2] is used (e.g., from the *Geant* network in [23]), it is rather old and mostly from research networks which do not feature the same characteristics as large ISP backbones. Hence, it is unclear whether the results obtained on such data are directly transferable to a practical application in large commercial networks. (iv) While some works (e.g., [7]) feature an extensive evaluation on a large set of different networks, others (e.g., [23] or [26]) only test their approaches on a very limited number of networks (6 and 2, respectively). Even though their results look promising, the sample size is probably far to low to allow for a meaningful generalization of the results to other networks. And lastly, (v) while some approaches (e.g., DP) sound very promising in theory, there are no evaluation results reported in the literature, at all.

We aim to address the above issues by carrying out an extensive performance evaluation of all preprocessing approaches on a large set of networks from the Repetita dataset as well as recent network data from a globally operating Tier-1 ISP.

## IV. Evaluation Setup

This section presents our evaluation setup by introducing the used datasets and describing the respective algorithm implementations. All computations are carried out on the same 64-core 3.3GHz machine with around 500GB of RAM and using CPLEX 20.1.0 [14] as LP-solver.

### A. Data

We carry out our evaluation on two sets of data. The first one consists of data from the publicly available *Repetita* dataset [11]. It features topologies of real-world networks (mostly WANs or ISP backbones) collected in the *Internet Topology Zoo* [16] and artificially generated traffic matrices (using a *random gravity model* [19]) for each topology. In addition to that, each topology also comes with two sets of IGP metrics (*unary* and *inverse capacity*). However, we limit our evaluations to only the *unary* metric set as previous results [22] have shown that the impact of different metric designs on SR performance is negligible. Other results also indicate that SR middlepoint selection approaches also seem to be quite robust regarding the underlying metric (cf. e.g., [23]). We further discard all instances already solved optimally by SPR. Finally, since for smaller networks with just a couple tens of nodes, even rather complex LPs are generally solvable within seconds or less (cf. e.g., [5]), there is basically no practically relevant improvement to achieve for these networks. Therefore, we limit our evaluations to larger networks with at least 40 nodes. This leaves us with a total of 72 networks comprising of 40 to 197 nodes and around 85 to 500 edges (cf. Table I).

Complementary to the Repetita data with artificial traffic, we also carry out evaluations on a second set of data collected from the backbone network of a globally operation Tier-1 ISP. It features 19 topology snapshots that resemble different expansion states of the network between 2017 and 2021 and a real traffic-matrix collected during the peak-hour of the respective day. Table I lists some further information on the most important graph properties across the respective topologies in each of the two dataset used in our evaluation. Regarding the number of edges, parallel links are counted as just one edge and the density characterizes the ratio of (non-parallel) edges in the graph relative to a complete graph.

### B. Algorithms & Implementations

We limit our evaluation of the centrality-based middlepoint selection approaches to the GSP centrality as it was identified as performing best in previous works (cf. Section III-A). To get around the issues regarding the high algorithmic complexity of its computation [26] (and the resulting high computation times), we use an approximation algorithm provided by *NetworKit*[3] which approximates the node group with maximum centrality up to a given accuracy $\epsilon$. Such an approximation would (most likely) also be used in a practical deployment due to the substantial performance gains. For our evaluations, we use $\epsilon = 0.005$ which allows to compute the respective maximum centrality group in a couple of seconds for most instances. We implement the SB approach as described in Section III-B, with a small extension to address the already

[2]Meaning topology *and* traffic data obtained from real operational networks.

[3]https://networkit.github.io/

Table I: Graph properties of the topologies in the two datasets used for evaluation.

| | Repetita (72 Topologies) | | | | ISP (19 Topologies) | | | |
|---|---|---|---|---|---|---|---|---|
| | min | max | avg | stdDev | min | max | avg | stdDev |
| Nodes | 40 | 197 | 68.69 | 31.81 | 108 | 186 | 143.11 | 29.90 |
| Edges | 86 | 486 | 171.94 | 77.31 | 660 | 1064 | 897.16 | 136.25 |
| Density [%] | 1.26 | 7.82 | 4.30 | 1.48 | 3.09 | 6.57 | 4.73 | 1.35 |
| Diameter | 4 | 35 | 11.79 | 7.57 | 6 | 8 | 7.32 | 0.58 |

described issues regarding demands with very low shortest path lengths. For this, we allow each demand with a shortest path length of just one hop to be rerouted over arbitrary paths with two hops (irrespective of the chosen $\alpha$-value). This turns out to be sufficient to resolve most of the respective issues without significantly increasing the overall number of SR paths to consider during optimization. The same was also observed in [7]. To implement DP, we first sort all traffic demands by size in ascending order. After this, we keep fixing the smallest demands to their SPR paths until the total sum of "fixed" traffic reaches a certain share of the total traffic volume given by the parameter $\alpha_{DP} \in [0, 1]$.

Similar to the related work, we evaluate the effectiveness of the preprocessing approaches based on the 2SR LP (Problem 1). It is the de-facto standard LP for SR TE and builds the foundations for a wide body of derivative work (cf. Section II) to which the findings should be transferable. Our evaluation focuses on the resulting MLU deterioration and the achievable speedup compared to the standard 2SR implementation.

## V. Evaluation Results

In this section, we evaluate the performance of the various middlepoint selection approaches presented in Section III. The MLU deteriorations and the achievable speedup for different parameterizations are depicted in Figure 1 and 2, respectively, with individual subfigures for each approach. Orange boxplots show the respective distributions for the Tier-1 ISP dataset and blue boxplots for the Repetita dataset. In the context of the following evaluation, the *speedup factor* is used to characterize the performance improvements regarding computation time achievable with the different preprocessing approaches. It is calculated by dividing the computation time $T_{default}$ required by the default algorithm (without any preprocessing) by the computation time of the same algorithm $\mathcal{A}$ when the respective preprocessing approach $p$ is applied beforehand (*including* the computation time of the respective preprocessing):

$$SpeedupFactor(p, A) = \frac{T_{default}(\mathcal{A})}{T(p(\mathcal{A}))} \tag{8}$$

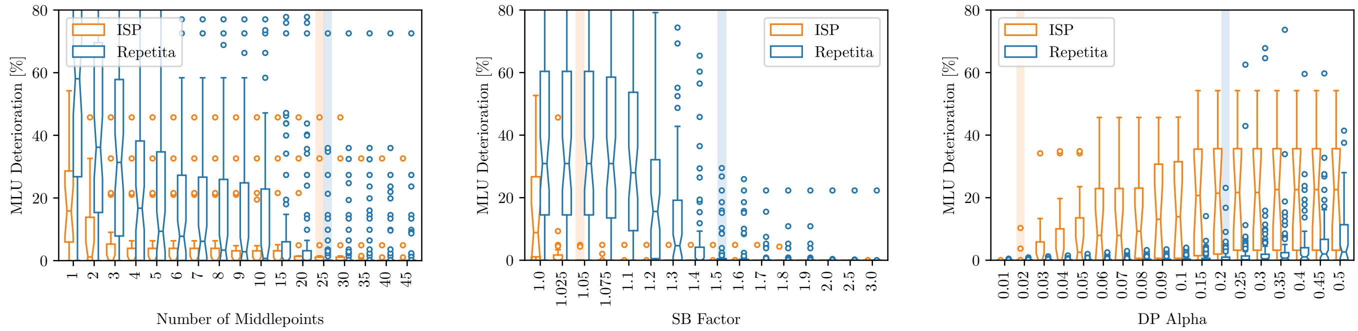
### A. A Primer Regarding CPLEX-Related Outliers

In rare occasions, there can be outliers with a speedup factor below one (e.g., in Figure 2c for $\alpha_{DP} = 0.3$ for the ISP data). This means using the respective preprocessing approach actually resulted in an increase in computation time. While this is rather surprising at first thought, there is a rather simple explanation for this phenomenon. LP-solvers like CPLEX stop optimization only if they find a "proof" that the currently best solution is truly optimal (or within a small margin to the optimum). This *optimality gap* is computed by comparing the currently best found solution against a lower bound for the best possible objective value which is continuously updated (increased) during optimization. If the gap between the lower bound and the best found solution is sufficiently small, the solution is considered to be optimal. By preemptively limiting the allowed set of available SR paths, the rare scenario can occur in which we prohibit a path that might not be required for an optimal solution but that facilitates a quick proof of optimality. There might be other options for such a proof but if these are explored in a much later stage of the *branch-and-cut search*, proving optimality and, thus, the whole optimization process might take substantially longer. The same effect is also responsible for the more noticeable "drop" of the achievable speedup when going from 10 to a single digit number of middlepoints for the ISP dataset in Figure 2a. Normally, over all preprocessing approaches, the achievable speedup increases when reducing the set of available SR paths. The same can be observed here from 45 to just 10 available middlepoints. However, when only allowing 9 or less middlepoints, the speedup unexpectedly drops instead of increasing further. We suspect that the reason for this is the exclusion of a node (and, thus, an SR path) that enables a fast proof of optimality. However, this node (or the related SR paths) do not seem to have an impact on the solution quality as the MLUs for 9 and 10 middlepoints are basically identical (cf. Figure 1a).

While the possibility of actually degrading performance when applying preprocessing approaches might be concerning, there is a straight-forward solution. Increasing the allowed *optimality gap* of CPLEX by only a small amount allows for an easier proof of "optimality" (even without the paths excluded by the preprocessing). In our experiments, increasing the optimality gap from the default $10^{-4}$ to around $10^{-3}$ proved promising to resolve these issues without having a practically relevant negative impact on the solution quality. Solutions are still within $0.1\%$ (instead of $0.01\%$) of the optimum. In the context of practical deployments, such minute differences are basically negligible since traffic, while mostly being quite stable, is still subject to small ongoing variations. Those (most likely) cover up such marginal MLU differences.

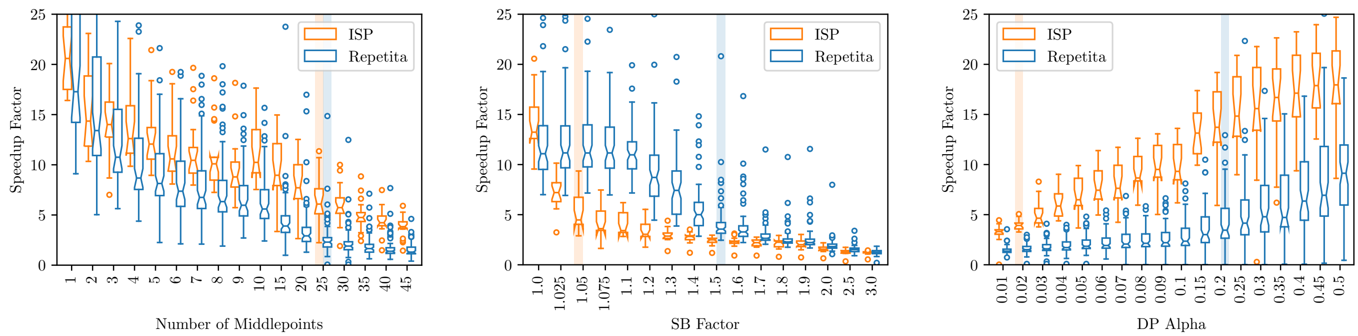
### B. Centrality-based Middlepoint Selection

For the centrality-based middlepoint selection (Figures 1a and 2a), it can be seen that allowing only a small set of nodes as available middlepoints can result in a substantial (factor 10–20) speed-up in computation time. This, however, comes




(a) Centrality-based (b) Stretch Bounding (extended) (c) Demand Pinning

Figure 1: MLU deterioration for different preprocessing approaches. (A few very large outliers were cut off for better readability.)




(a) Centrality-based (b) Stretch Bounding (extended) (c) Demand Pinning

Figure 2: Achievable speedup for the 2SR optimization. (A few very large outliers were cut off for better readability.)

at the price of a significant deterioration of the overall solution quality. In the worst cases, MLUs increase by up to 55% for the ISP dataset and by more than 80% for the Repetita instances. By increasing the number of allowed middlepoints, these MLU deteriorations can be reduced but this also results in an increase in computation time. Ultimately, operators have to make an individual decision regarding the acceptable trade-off between speedup and resulting MLU deterioration. This might vary for different use cases, but from our experience, the highest acceptable MLU deterioration for most scenarios probably lies somewhere around 10-15%, at most. Based on this, we highlight in each plot the parameter configuration that produces the highest speedup while still allowing for, in our experience, practically usable MLUs. For the Repetita data this is around 25 middlepoints. For the ISP instances, the box and whiskers are already below the 10% deterioration threshold for just three middlepoints. However, since the dataset only comprises 19 instances, the three "outliers" that (considerably) surpass this threshold still make up over 15% of the dataset. To improve the solution quality for these instances, substantially higher numbers of middlepoints are needed (in the range of 25-45). Hence, we argue that the number of middlepoints required to obtain practically usable solutions is (to some extend) instance-dependent but, in general, at least 20 to 25 middlepoints seem to be required. This translates to an average speedup factor of around 3-4 for the Repetita dataset and 7-8 for the ISP backbone (cf. Figure 2a). It has to be noted, however, that for these parameterizations there is still a considerable number of instances with a significant MLU deterioration left. Overall, our results generally confirm the findings of [26]. Selecting only a few central nodes as available middlepoints can substantially reduce computation times but also results in significant deteriorations of the overall MLUs, especially for small numbers of middlepoints.

### C. *Stretch-Bounding*

In Figure 1b and 2b, it can be seen that, for the ISP dataset, near optimal results are achieved with a SB factor of just 1.05. For a factor of 1.1, there is basically no noticeable MLU deterioration anymore while still achieving a speedup of factor 4 to 5. For the Repetita dataset, the results differ noticeably. Here, such low SB factors result in a substantial MLU deterioration of around 40% on average and over 80% at max. Practically usable results can be achieved with a SB factor of 1.5 or higher and (virtual) optimal results require a factor of around 2.0. This translates to a speedup of around factor 4.5 and factor 2, respectively.

At first glance, it might seem like the SB approach performs substantially worse for the Repetita dataset. This observation, however, is (at least a bit) deceptive. While, for low SB factors, the MLU deterioration on the Repetita dataset is substantially higher, the speedup is also much better. The reason for this is that for the same SB factor, the overall number of prohibited SR paths is much higher for the Repetita dataset. For example, for a SB factor of $1.1$, around 90% or more of all available SR paths are prohibited for many Repetita instances. Contrary, for the ISP data, only around 65-70% of paths are filtered out. As a result, the optimization for the Repetita instances is faster due to the lower number of options to evaluate, but this also results in a worse overall solution quality. If we instead compare results based on the percentage of excluded SR paths, they become much more similar. For example, for a SB factor of $1.4$, the percentage of excluded SR paths is also in the range of 65-70% and the resulting speedup is comparable to the one of the ISP data with the respective "matching" SB factor of $1.1$. We suspect that these variations are a product of topological differences between the instances in the Repetita dataset and the real ISP backbone network. However, investigating and identifying these differences is out of the scope of this work, but remains an interesting question for future work.

### *D. Demand Pinning*

Results for the DP approach are depicted in Figures 1c and 2c. It can be seen that for the ISP data, a speedup of around factor $4.5$ can be achieved without substantially worsening the MLU. However, for larger $\alpha$-values, the solution quality deteriorates quickly. Results for the Repetita data look rather different. Here, we are able to exclude up to 20% and more of the total traffic volume before a relevant MLU deterioration becomes observable. However, the speedup, while overall slightly better than for the ISP data, remains rather similar with a factor of around 5. The reason for those differences lies in the distribution of the demand sizes in the traffic matrices of the two datasets. The real ISP traffic features a substantially higher number of really small demands (w.r.t. the total traffic) than the artificially generated matrices in the Repetita dataset. This is exemplarily depicted in Figure 3 for the largest instance of the ISP and Repetita dataset, respectively. As a result, the same $\alpha$-value allows for the exclusion of substantially more demands for the ISP network. An $\alpha$-value of $0.01$, for example, excludes around 70-80% of all demands in the ISP matrices from optimization, while only excluding around 15-20% of demands from the Repetita matrices.

### *E. SR Path Domination*

Figure 4 depicts the speedup that is achievable with the SR path domination approach on our two datasets. Contrary to the previous approaches, there is no need to look at MLU deterioration since the main idea of the SR path domination concept is to retain provable optimality of the achievable MLUs. It can be seen that, for the Repetita data, computation times can be improved by around a factor of four on most instances with a couple of outliers even reaching close to factor 10. Those high outliers are a result of the special topology structures of certain instances. For example, the `Ulaknet` topology consists of three star shaped networks whose centers are interconnected with each other. Basically all SR paths

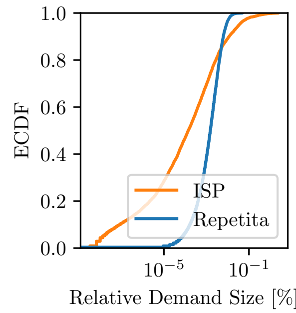



Figure 3: ECDF of the relative demand sizes (w.r.t to the total traffic volume) of the ISP `2021` and the Repetita `Cogentco` instances.

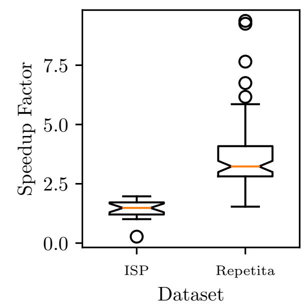


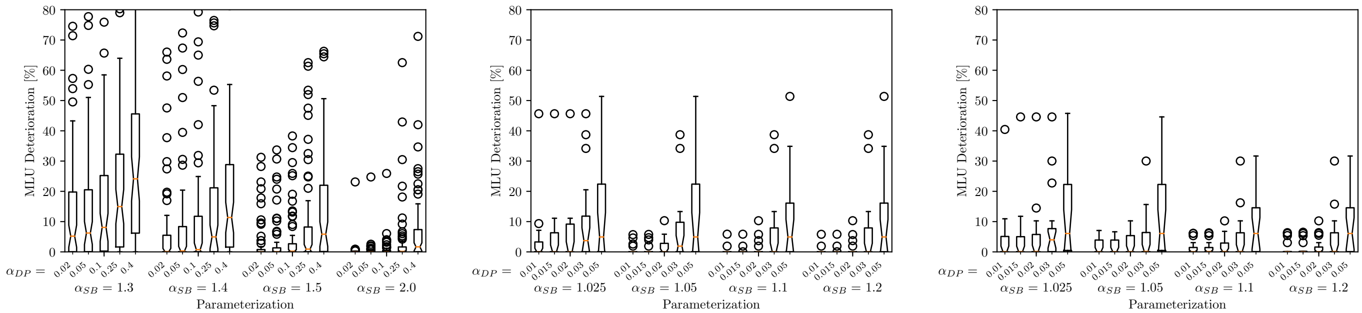

Figure 4: 2SR speedup achieved by the *SR path domination* preprocessing on the two evaluation datasets.

using one of the many stub-nodes as intermediate segment provide no TE benefit regarding the MLU and can be ignored. This results in over 98% of all SR paths being excluded from optimization. On more "realistically" shaped topologies (w.r.t. common network design principles), however, this number is much lower (mostly between 65-80%) and, hence, the achievable speedup is also more moderate.

While the SR path domination preprocessing works quite well for the Repetita data, this does not hold for the real-world ISP network. Here, the average achievable speedup factor is just around 1.5 and the maximum barely surpasses factor 2. The reason for this, again, lies in the number of SR paths that are ruled out for each respective dataset. For the ISP dataset, this number is substantially lower with just around 20% of the total number of SR paths compared to an average of around 80% for most Repetita instances. We do not have a definitive answer what causes this behavior but we suspect that it is a result of topological differences between the networks in the Repetita dataset and the real ISP network. For example, the ISP network has virtually no stub-nodes since a common design goal for modern networks is to achieve at least two-connectivity for all nodes. This facilitates reliability and robustness as it ensures that the network will not be partitioned by single-link failures. Contrary, the Repetita topologies feature a rather large number of stub-nodes. Since those are never needed as middlepoints to obtain an optimal solution, a larger number of stub nodes automatically results in a larger number of dominated SR paths. This becomes visible when removing all stub-nodes from the Repetita topologies which reduces the number of excluded SR paths from around 80% on average to just 60%. For reasons of space, we cannot delve deeper into this topic here and leave it for future work.

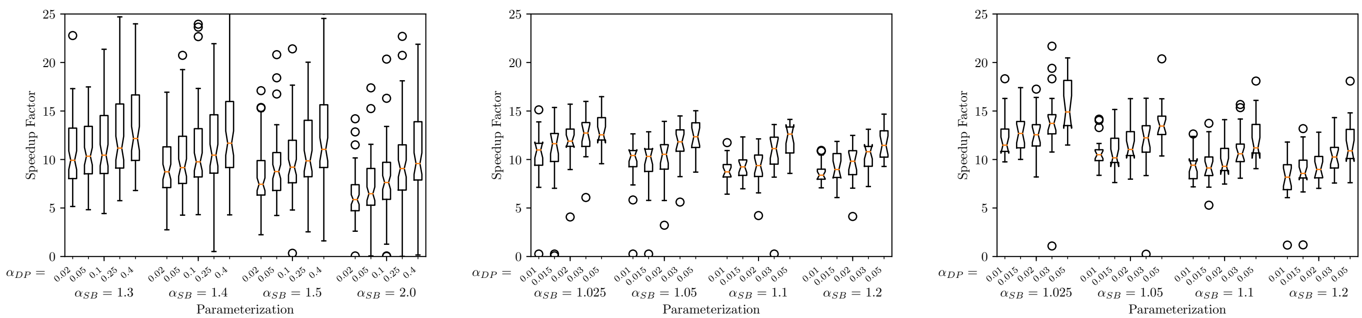
### *F. Discussion*

All in all, we have seen that each preprocessing approach has its pros and cons. Some perform better on the Repetita data and some on the ISP data. Hence, there is no clear "winner" to be picked but we hope that our extensive analysis facilitates others in picking a suitable preprocessing approach for their




(a) 2SR on Repetita dataset (b) 2SR on ISP dataset (c) SC2SR on ISP dataset

Figure 5: MLU deterioration resulting from the combined preprocessing approach for different datasets and SR algorithms.




(a) 2SR on Repetita dataset (b) 2SR on ISP dataset (c) SC2SR on ISP dataset

Figure 6: Speedup achieved by the combined preprocessing approach for different datasets and SR algorithms.

applications. However, what became more and more clear during our evaluation, is the fact that considerable differences in results are observable depending on the dataset used. This especially holds true for the SR path domination approach that works really well for Repetita networks but considerably worse for the real Tier-1 ISP backbone. We track this down to differences in topology and traffic characteristics between the two datasets. This reinforces our concerns regarding the direct transferability of results obtained on the Repetita data with artificial traffic to real networks, already expressed in Section III-E. It also stresses the importance of also carrying out evaluations on real, recent network data. Of course, our ISP dataset is "just one datapoint" that does not allow to draw definitive and universal conclusions but other recently reported information e.g., regarding the demand size distribution in the *Microsoft* network [17] is far closer to the ISP network characteristics than to the Repetita data. To further investigate this, it would be desirable to repeat our experiments on other recent data from real networks. However, to the best of our knowledge, there currently are no publicly available datasets that provide such information.

## VI. Combining Preprocessing Approaches to Improve Performance

As seen before, there is no definitive answer to what the generally best preprocessing approach is since performance varies between datasets. In this section we propose a concept for combining multiple preprocessing approaches to allow for a more consistent performance across all datasets and to further improve the achievable speedup.[4]

### A. Concept

Our approach is based on the observation that, until now, we always considered each preprocessing approach individually. However, they are not mutually exclusive. Therefore, it is possible to combine the different preprocessing approaches into a single one. This can yield multiple benefits. First and foremost, it holds the potential of further increasing the achievable speedup. Combining the individual sets of excluded SR paths allows to further increase the number of SR paths that can be ignored during optimization. However, it is unclear whether the combination of multiple exclusion sets that perform well individually will result in a well performing union set, as well. The combined set might also become too restrictive and, hence, might result in substantial MLU deterioration. Secondly, combining different preprocessing approaches might also lead to a more "stable" performance across our two datasets. In simple terms, by combining an approach that works better on the ISP data (i.e., SB) with one that seems more suited for the Repetita data (i.e., SR path domination), we hope to leverage their individual benefits and get an algorithm that performs well on both datasets.

[4] A conceptually related but less extensive (and, thus, less effective) approach is also considered in [7], showing promising preliminary results.

Table II: Computation times (in seconds) of the respective ground truth algorithms without any preprocessing.

| | | Min | Max | Median | Average |
|---|---|---|---|---|---|
| 2SR | Repetita | 4 | 5456 | 25 | 248 |
| | ISP | 178 | 1407 | 691 | 611 |
| SC2SR | ISP | 2477 | 14568 | 5604 | 6571 |

For our improved preprocessing algorithms, we combine the three approaches of SB, DP and SR path domination. The reason for not including the centrality-based approach is that it generally performs worse than the other approaches with regards to MLU deterioration and speedup. Furthermore, as already discussed in Section III-A, it also features other weaknesses when it comes to practical use (e.g., regarding latency constraints). Our new combined preprocessing approach starts with a DP operation that can be fine-tuned with the $\alpha_{DP}$ parameter. After this, a SB step is carried out using the $\alpha_{SB}$ parameter. The SR path domination filtering comes last as it is the computationally most demanding operation. Having already filtered out a large set of SR paths by the previous two operations which do not need to be checked for domination or equivalency anymore, facilitates lower computation times.

### *B. Evaluation*

We evaluate the performance of our combined approach on our two evaluation datasets for the 2SR algorithm. Additionally, we also carry out a short exemplary evaluation for the MO-capable SC2SR algorithm proposed in [2]. This is done to provide an insight into whether results are transferable to other SR TE algorithms even if those utilize a rather different SR variation. The results regarding MLU deterioration and speedup for various parameter combinations are depicted in Figures 5 and 6, respectively. It can be seen that for the Repetita dataset (Figures 5a and 6a), with the right parameter configuration, we are able to achieve a 2SR speedup of around factor 7 to 8 without a significant MLU deterioration. If a few more outliers are acceptable, this factor can be increased even further close to a ten-times speedup. The same applies to the ISP dataset for both the 2SR (Figures 5b and 6b) and the MO-capable SC2SR algorithm (Figures 5c and 6c). This confirms that the new preprocessing approach performs (more or less) equally good across both datasets when the right parameters are chosen. Furthermore, it also shows that the preprocessing is transferable to other SR optimization algorithms. It achieves comparable results regarding MLU deterioration and speedup, even if the underlying SR concept is inherently different.

To put into perspective what a speedup of around factor 10 actually means, some information on the computation times of the ground-truth algorithms (without any preprocessing) are given in Table II. It can be seen that, for example, the SC2SR algorithm takes around two hours to compute, on average, and over four hours at max. With our preprocessing, this can be reduced to just around 10 or 20 minutes, respectively. The benefits of our preprocessing become even more apparent when looking at the 2SR algorithm. Here, our preprocessing is able to reduce the average computation time from 10min and more, to less than 2min for most of the ISP instances. This easily allows for the use of LP-based optimization for use cases where network configuration is continuously adapted on a timescale of just a few minutes (cf. e.g., [13]). Furthermore, we are now advancing into computation time regions in which it can be argued that the 2SR algorithm could even be used for tactical TE that allows to quickly react to failures or traffic shifts [3]. Finally, we also believe that the performance achieved with our preprocessing approach is reasonably close to what can actually be achieved with preprocessing in general. The reason for this are its extremely high numbers of excluded SR paths. For the 2SR algorithm, our preprocessing already rules out 97-99% of all theoretically configurable 2SR paths. This probably does not leave much room for further improvement since a certain number of options to choose from is required before solution quality starts to degrade substantially.

## VII. Discussion

In our study, we only consider SR using at most two node segments per path. While it has been shown that this is sufficient to obtain virtually optimal solutions in many practical use cases [1], [20], there also are scenarios in which higher order segment paths or the use of adjacency segments can be necessary. We argue that the performance gains achievable with preprocessing approaches for such algorithms are probably even higher than those observed by us regarding 2SR. The reasoning for this is as follows. While adjacency segments or a general higher number of segments can be required to facilitate the implementation of certain forwarding paths that cannot be build with 2SR, this number is relatively small in most scenarios. Most of the newly considered paths can already be implemented with 2SR or do not have any practical value (i.e. looping paths that visit a segment multiple times). Thus, increasing the number of segments generally also results in an increase in the ratio of “useless to useful” paths, which, in turn, improves the effectiveness of preprocessing approaches. This has also been reported by Callebaut et al. [7]. They show that the number of dominated SR paths grows from only 50% when using 2SR to around 90% and 97% for 3SR and 4SR, respectively. Thus, without having explicitly considered higher order segment paths or adjacency segments in our study, the performance improvements shown here should resemble a lower bound for what can be achieved for those, as well.

Furthermore, our study (and other publications on the topic of SR preprocessing) only consider the performance benefits achievable for LP-based approaches, since those notoriously suffer from scalability issues, negatively impacting their usability for large networks. However, the observation that large portions of SR paths can be ignored during optimization without considerably worsening the achievable solution quality could also find applications in other contexts. Tactical TE, for example, focuses on the computation of reasonably good TE configurations within very strict time constraints in order to facilitate fast reconfiguration in the presence of failures or other critical events. The ability to preemptively limit the explored solution space with preprocessing approaches looks promising to further speed up such heuristic computations as well, especially since optimality of a solution is not strictly required anyway, as long as it is able to resolve the critical events. However, this would probably require faster and more efficient preprocessing approaches since, for the largest networks, our combined preprocessing takes around a minute to compute. In the context of LP-based optimization that, without the preprocessing, can take multiple hours, this is well worth it. However, in the context of tactical TE where solutions have to be computed within seconds or at most a couple of minutes, spending a considerable amount of time on just the preprocessing is probably not feasible. Here, faster and more efficient preprocessing approaches are needed.

# VIII. Conclusion

In this paper, we conducted the first large scale comparative study of existing preprocessing (or *middlepoint selection*) approaches for SR. For this, we not only used publicly available data from the *Repetita* dataset, but also real network data from a globally operating Tier-1 ISP. Based on the insights gained from this study, we proposed a combination of multiple preprocessing approaches to further improve performance. With this approach, the number of 2SR paths to consider for optimization can often be preemptively reduced by as much as 97-99%, while still obtaining close to optimal solutions. This lowers the computation times of different LP-based TE algorithms by a factor of 10 or more without a significant deterioration in solution quality. This represents a major improvement over the current state-of-the-art and facilitates the reliable use of LP-based TE in large segment-routed networks.

Similar to the literature, we focused on performance gains for LP-based algorithms in this paper. However, preprocessing approaches could also provide benefits in other contexts, as well. On of those being heuristic algorithms often used for fast, tactical TE [3], [10] where low computations times are crucially important. Here, preemptively limiting the explored solution space by excluding a large set of (non beneficial) SR paths, holds the potential to facilitate major performance gains. However, this most likely requires faster and more efficient ways to compute the respective preprocessing in order to keep up with the tight time constraints. We plan to look into this in the future.

# References


[1] R. Bhatia, F. Hao, M. Kodialam, and T. V. Lakshman, "Optimized Network Traffic Engineering using Segment Routing," in *Proc. of the IEEE Int. Conf. on Computer Communications (INFOCOM)*, 2015, pp. 657–665.

[2] A. Brundiers, T. Schüller, and N. Aschenbruck, "Midpoint Optimization for Segment Routing," in *Proc. of the IEEE Int. Conf. on Computer Communications (INFOCOM)*, 2022, pp. 1579–1588.

[3] ——, "Tactical Traffic Engineering with Segment Routing Midpoint Optimization," in *Proc. of the IFIP Netw. Conf. (NETWORKING)*, 2023.

[4] ——, "On the Benefits of Loops for Segment Routing Traffic Engineering," in *Proc. of the IEEE Conf. on Local Computer Networks (LCN)*, 2021, pp. 32–40.

[5] ——, "Combining Midpoint Optimization and Conventional End-to-End Segment Routing for Traffic Engineering," in *Proc. of the IEEE Conf. on Local Computer Networks (LCN)*, 2023, pp. 1–9.

[6] ——, "An Extended Look at Midpoint Optimization for Segment Routing," *IEEE Open Journal of the Communications Society*, vol. 5, pp. 1447–1468, 2024.

[7] H. Callebaut, J. De Boeck, and B. Fortz, "Preprocessing for segment routing optimization," *Networks*, pp. 1–20, 2023.

[8] M. G. Everett and S. P. Borgatti, "The Centrality of Groups and Classes," *Journal of Mathematical Sociology*, pp. 181–201, 1999.

[9] C. Filsfils, N. K. Nainar, C. Pignataro, J. C. Cardona, and P. Francois, "The Segment Routing Architecture," in *Proc. of the IEEE Global Communications Conf. (GLOBECOM)*, 2015.

[10] S. Gay, R. Hartert, and S. Vissicchio, "Expect the Unexpected: Sub-Second Optimization for Segment Routing," in *Proc. of the IEEE Int. Conf. on Computer Communications (INFOCOM)*, 2017.

[11] S. Gay, P. Schaus, and S. Vissicchio, "REPETITA: Repeatable Experiments for Performance Evaluation of Traffic-Engineering Algorithms," *ArXiv e-prints*, 2017.

[12] R. Hartert, S. Vissicchio, P. Schaus, O. Bonaventure, C. Filsfils, T. Telkamp, and P. Francois, "A Declarative and Expressive Approach to Control Forwarding Paths in Carrier-Grade Networks," in *Proc. of the ACM Conf. on Special Interest Group on Data Communication (SIGCOMM)*, 2015, pp. 15–28.

[13] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving High Utilization with Software-Driven WAN," in *Proc. of the ACM Conf. on Special Interest Group on Data Communication (SIGCOMM)*, 2013, pp. 15–26.

[14] IBM, "IBM ILOG CPLEX Optimization Studio 20.1.0," https://www.ibm.com/docs/en/icos/20.1.0, 2020.

[15] M. Jadin, F. Aubry, P. Schaus, and O. Bonaventure, "CG4SR: Near Optimal Traffic Engineering for Segment Routing with Column Generation," in *Proc. of the IEEE Int. Conf. on Computer Communications (INFOCOM)*, 2019, pp. 1333–1341.

[16] S. Knight, H. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The Internet Topology Zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, 2011.

[17] U. Krishnaswamy, R. Singh, N. Bjørner, and H. Raj, "Decentralized Cloud Wide-Area Network Traffic Engineering with BlastShield," https://www.microsoft.com/en-us/research/uploads/prod/2021/11/blastshield-61a6c89e0ca05.pdf, 2021, Microsoft.

[18] P. Namyar, B. Arzani, R. Beckett, S. Segarra, H. Raj, and S. Kandula, "Minding the Gap between Fast Heuristics and Their Optimal Counterparts," in *Proc. of the ACM Workshop on Hot Topics in Networks (HotNets)*, 2022, pp. 138–144.

[19] M. Roughan, "Simplifying the Synthesis of Internet Traffic Matrices," *SIGCOMM Comput. Commun. Rev.*, pp. 93–96, 2005.

[20] T. Schüller, N. Aschenbruck, M. Chimani, M. Horneffer, and S. Schnitter, "Traffic Engineering using Segment Routing and Considering Requirements of a Carrier IP Network," *IEEE/ACM Transactions on Networking*, vol. 26, no. 4, pp. 1851–1864, 2018.

[21] T. Schüller, N. Aschenbruck, M. Chimani, and M. Horneffer, "Failure Resiliency With Only a Few Tunnels – Enabling Segment Routing for Traffic Engineering," *IEEE/ACM Transactions on Networking*, vol. 29, no. 1, pp. 262–274, 2021.

[22] T. Schüller, N. Aschenbruck, M. Chimani, and M. Horneffer, "On the Practical Irrelevance of Metrics on Segment Routing Traffic Engineering optimization," in *Proc. of the IEEE Conf. on Local Computer Networks (LCN)*, 2018, pp. 640–647.

[23] T. Settawatcharawanit, Y.-H. Chiang, V. Suppakitpaisarn, and Y. Ji, "A Computation-Efficient Approach for Segment Routing Traffic Engineering," *IEEE Access*, pp. 160 408–160 417, 2019.

[24] T. Settawatcharawanit, V. Suppakitpaisarn, S. Yamada, and Y. Ji, "Segment Routed Traffic Engineering with Bounded Stretch in Software-Defined Networks," in *Proc. of the IEEE Conf. on Local Computer Networks (LCN)*, 2018, pp. 477–480.

[25] G. Trimponias, Y. Xiao, X. Wu, H. Xu, and Y. Geng, "Node-Constrained Traffic Engineering: Theory and Applications," *IEEE/ACM Transactions on Networking*, vol. 27, no. 4, pp. 1344–1358, 2019.

[26] G. Trimponias, Y. Xiao, H. Xu, X. Wu, and Y. Geng, "Centrality-based Middlepoint Selection for Traffic Engineering with Segment Routing," *ArXiv e-prints*, 2019.

[27] P. L. Ventre, S. Salsano, M. Polverini, A. Cianfrani, A. Abdelsalam, C. Filsfils, P. Camarillo, and F. Clad, "Segment Routing: A Comprehensive Survey of Research Activities, Standardization Efforts, and Implementation Results," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 1, pp. 182–221, 2021.